Applying Computational Intelligence Techniques to QoS Time Series Forecasting in Services Computing

Yang Syu[1*], Yong-Yi FanJiang[2]
[1]Institute of Information Science, Academia Sinica, Taipei City, Taiwan.
[2]Department of Computer Science and Information Engineering, Fu Jen Catholic University, New Taipei City, Taiwan
* Corresponding author. Email: yyfanj@csie.fju.edu.tw
Manuscript submitted May 10, 2018; accepted July 8, 2018.
doi: 10.17706/jcp.13.9.1010-1026

Abstract: Recently, the forecasting of dynamic Quality of Service (QoS) values for Web Services (WSs) has become an emerging topic in services computing. In most previous research, various time series forecasting methods have been used to address this problem. In this paper, we propose the use of two computational intelligence techniques, namely, genetic programming (GP) and support vector regression (SVR). To demonstrate the forecasting performance of the two proposed techniques, we compare them with the conventional methods based on experiments run on a real-world dynamic QoS time series dataset. Our experimental results show that the proposed GP and SVR methods outperform the conventional methods in both training (in-sample) and testing (out-of-sample) accuracy. Between the two proposed approaches, we find that GP might be the better choice overall. In terms of training performance, GP is superior in terms of both the individual and average experimental results; however, this is not the case for testing performance. In terms of testing accuracy, SVR outperforms GP in many individual experiments; however, SVR also yields extremely poor forecasting accuracy in several individual experiments, indicating that it is unstable and unreliable. In many of the individual experiments, GP is only insignificantly inferior to SVR, and it still achieves the best average forecasting accuracy according to two of the three considered measures.

Key words: Time series forecasting, quality of service, computational intelligence techniques, support vector regression, genetic programming.

1. Introduction

The QoS attributes of a *Web Services* (WS) can be broadly divided into two categories, namely, static QoS attributes and dynamic QoS attributes. The former are QoS attributes with fixed values (e.g., price), whereas the latter are attributes whose values vary over time (e.g., response time and availability). Dynamic QoS attributes are the focus of this study. The real examples of the variations in the dynamic QoS attributes of WSs (dynamic QoS time series) can be found in Section II of [1], Section 3.2 of [2], Section 2 of [3], Section 3.3 of [4], Section 8 of [5], and Section III.B of [6].

In services computing, predicting the values of a WS's dynamic QoS attributes based on historical information (i.e., past QoS observations) is an important emerging topic because the static QoS information provided by service providers is often unrealistic and imprecise [7]. Additional motivations for the advance prediction of dynamic QoS values and the benefits and applications of such forecasting approaches, such as for the proactive detection and replacement of QoS-violating WSs, can be found in the literature, which we

review and discuss in Section 2.

To address the problem of predicting dynamic QoS values, as shown in Section 2 of this paper, most previous research has treated the problem as one of time series forecasting, and various statistics-based methods for time series forecasting have been employed, such as Auto-Regressive Integrated Moving Average (ARIMA) models and Exponential Smoothing (ES) methods. In this paper, instead of using traditional time series forecasting methods, we propose to address the problem using computational intelligence techniques, such as *genetic programming* (GP) and *support vector regression* (SVR). More specifically, we use GP and SVR to seek a predictor that can produce accurate dynamic QoS predictions.

The manner in which prediction is performed using the adopted computational intelligence techniques (i.e., the form of the predictor and the way in which it is generated) differs from the approach used in traditional time series methods; indeed, computational intelligence techniques are quite innovative compared with traditional time series methods, which have a long history of development based on statistical techniques. The hope is to take advantage of their unique features and advantages (which are explained in detail in Section 4 of this paper) to achieve better forecasting performance (i.e., lower forecasting error and higher prediction accuracy). Another motivation for investigating the use of computational intelligence techniques for this purpose is that they have been successfully applied for time series forecasting in other application domains; for example, GP has been applied to financial time series in [8]-[11] and for supply chain management in [12].

In this study, we also consider several conventional methods for comparison, including the two current de facto solutions for time series forecasting, i.e., ARIMA models and ES methods [13]. Before we describe the proposed computational intelligence techniques and the conventional methods with which they are to be compared, the problem to be solved and the accuracy metrics to be used are formally defined. Finally, for the evaluation of the considered approaches in a realistic scenario, we present a set of experiments run on a real-world dynamic QoS dataset and then analyze and discuss the experimental results in detail.

Below, we describe the contributions of this paper. First, to the best of our knowledge, we are the first to propose the application of these two computation intelligence techniques, i.e., GP and SVR, to solve the dynamic QoS forecasting problem. Based on our experimental results, we find that overall, the two proposed techniques outperform the traditional, well-developed time series methods in both training and testing forecasting accuracy. Second, because we include most representative time series approaches in our comparison, this study can serve as a guide for WS consumers when they wish to choose or develop a dynamic QoS forecasting solution for their own purposes. Finally, this paper comprehensively documents our study. All elements of the study, including the problem to be solved, the methods used to evaluate forecasting accuracy, and the proposed and conventional solutions are formally described and defined. Such definitions are partially or completely lacking in most existing works. However, without a formal mathematical definition of the considered problem, for example, many specifics remain unknown, thereby preventing further evaluations using the same problem specifications.

Our experimental results demonstrate that as a whole, the proposed GP and SVR methods are more accurate than the traditional methods. GP is shown to be the superior approach in terms of training accuracy (also called in-sample accuracy in time series forecasting [14]-[16]), which suggests that GP is the best choice for the modeling and fitting of known QoS time series. However, in terms of testing accuracy (out-of-sample accuracy), GP does not perform as well as it does during the training stage. With regard to testing accuracy, SVR outperforms GP on more than half of the forecasting problem instances considered in our experiments, but it also performs extremely poorly compared with GP for a few problem instances, which suggests that SVR might not be a stable and reliable means of solving the problem of interest. Moreover, in terms of average testing performance, GP is still superior to SVR according to two of the three

considered measures of forecasting accuracy.

The remainder of this paper is organized as follows. Section 2 reviews existing studies on dynamic QoS time series forecasting. The forecasting problem to be solved and the accuracy measures used in this paper are defined in Section 3. Subsequently, Section 4 and 5 introduce the proposed computational intelligence techniques and the conventional methods considered for comparison, respectively. Our experimental environment and results are fully described in Section 6, and the results are analyzed and discussed in detail in Section 7. Finally, Section 8 concludes the paper.

2. Related Work

This section reviews the existing research on dynamic QoS time series forecasting. Table 1 presents the reviewed studies; each study is described in the table in terms of its proposed forecasting approach and the methods considered for comparison. From Table 1, it is clear that most previous works have used various traditional time series methods to approach the problem, mostly ARIMA and ES methods, because it is intuitive to adopt such well-developed, fully elaborated methods based on traditional disciplines (in this case, statistics and time series forecasting) to address the problem of interest.

However, there are also a few works, i.e., [3] and [17], that have employed artificial neural networks (ANNs), another common computational intelligence (or machine learning) technique. Although the authors of [3] claimed that ANNs, with their flexibility and strong analysis capability, represent the strongest competition for traditional time series methods in solving the dynamic QoS time series forecasting problem, our experimental results indicate that ANNs are not as accurate as GP or SVR. The table also lists two exceptional cases in which neither conventional time series methods nor computational intelligence techniques were used, namely, [6] and [18]. The former addressed a three-dimensional QoS forecasting problem, which is not exactly identical to the problem targeted in this study, and the latter used structural equation modeling, which is rarely employed in time series forecasting.

In this paper, most of the time series methods presented in Table 1 are considered as methods for comparison; thus, their technical details can be found in Section 4 (ANNs) and Section 5.

Table 1. A List of Existing Dynamic QoS Time Series Forecasting Approaches and the Methods with which They have been Compared

Authors	Forecasting methods proposed/used	Forecasting methods considered for comparison
Zadeh and Seyyedi [3]	*ANNs*	*None*
Senivongse and Wongsawangpanich [17]	*ANNs*	*None*
Ye et al. [19]	*Multi-variate ARIMA and Holts-Winters (exponential smoothing)*	*Multi-variate ARIMA, Holts-Winters, and VAR*
Cavallo et al. [20]	--	*Average, last observation, linear regression, and ARIMA*
Amin et al. [1]	*ARIMA-GARCH*	*ARIMA*
Amin et al. [2]	*ARIMA/SETARMA*	*ARIMA*
Godse et al. [7]	*ARMA*	*None*
YunNi et al. [5]	*ARMA*	*None*
Yilei et al. [6]	*Tensor factorization*	*Three other matrix factorization approaches*
Mu et al. [18]	*Structural equation modeling*	*None*
Zia ur Rehman et al. [21]	--	*ARIMA and exponential smoothing*

3. Problem Formulation and Performance Measures

This section formally defines both the forecasting problem to be addressed and the accuracy measures to be used for performance evaluation.

3.1. Formal Problem Definition

Before formulating the problem, we first define the dynamic QoS time series of interest. In this paper, we consider the *response time* of a WS as our forecasting target because it has been widely used in many previous studies.

A dynamic QoS time series (TS) is defined as

$$QoS_{1...l} = \{q_1, q_2, ..., q_{l-1}, q_l\},$$

where l is the length of the TS, i.e., the number of (response time) observations it contains; $QoS_{1...l}$ denotes the set of WS response time observations recorded during the period $1, ..., l$; and q_i represents the response time of the WS at time i. Furthermore, a TS can be split into two segments for analysis, namely, a training sub-TS $TraQoS_{1...tral}$ and a testing sub-TS $TesQoS_{1...tesl}$:

$$QoS_{1...l} = \{TraQoS_{1...tral}, TesQoS_{1...tesl}\}$$

and

$$TraQoS_{1...tral} = \{tra\_q_1, tra\_q_2, ..., tra\_q_{tral-1}, tra\_q_{tral}\},$$

$$TesQoS_{1...tesl} = \{tes\_q_1, tes\_q_2, ..., tes\_q_{tesl-1}, tes\_q_{tesl}\},$$

where $tral$ and $tesl$ are the lengths of the training sub-TS and the testing sub-TS, respectively, and $tral + tesl = l$. Note that $tra\_q_1 = q_1$ and $tes\_q_{tesl} = q_l$ and that $tra\_q_{tral} = q_{tral}$ and $tes\_q_1 = q_{tral+1}$. The training sub-TS is used to generate a predictor and to calculate the training (in-sample) accuracy, and the corresponding testing sub-TS is used to measure the testing (out-of-sample) forecasting accuracy.

To generate a set of QoS forecasts, a forecasting approach must consist of two phases, i.e., *predictor generation* and *forecast generation*. **Predictor generation** can be simply expressed as follows:

$$QoSPredictor \leftarrow FA(TraQoS_{1...tral}),$$

where $QoSPredictor$ denotes the QoS (response time) predictor generated by forecasting approach FA with $TraQoS_{1...tral}$ as the available training data (i.e., the input to the algorithm). The forecasting approaches proposed in this study and the conventional approaches against which they are compared (namely, the various FA) are introduced in Section 4 and Section 5, respectively.

After a predictor is obtained, the algorithm proceeds to **forecast generation**. In this study, we consider one-step-ahead forecasting (a common assumption in current research); thus, to obtain a set of forecasts, the predictor must be iteratively applied to a set of predictor inputs that is constantly being updated. For example, the process of obtaining a set of QoS forecast values ($FTesQoS_{1...tesl}$) is expressed as follows:

$$ftes\_q_1 \leftarrow QoSPredictor(TraQoS_{1...tral}),$$

$$ftes\_q_2 \leftarrow QoSPredictor(TraQoS_{1...tral}, TesQoS_1),$$

$$.....$$

$$ftes\_q_{tesl} \leftarrow QoSPredictor(TraQoS_{1...tral}, TesQoS_{1...tesl-1}),$$

and

$$FTesQoS_{1...tesl} = \{ftes\_q_1, ftes\_q_2, ..., ftes\_q_{tesl}\}.$$

Forecasting is always performed for one step ahead; thus, at time i, the predictor can forecast the value at the one-step-ahead future time $i+1$, and all observations prior to time i (inclusive) are known and available.

Consequently, $\boldsymbol{FTesQoS_{1...tesl}}$ can be used to calculate the testing forecasting accuracy, as shown in the next section. Similarly, a set of training forecasts, $\boldsymbol{FTraQoS_{1...tral}}$, can be obtained in the same way for the calculation of the training accuracy.

3.2. Measures of Forecasting Accuracy

In this paper, both training (in-sample) accuracy and testing (out-of-sample) accuracy are considered. For both of them, the *mean absolute errors* (MAEs) and *mean absolute percentage errors* (MAPEs) of each forecasting approach are evaluated. In addition, for testing accuracy, we also consider the *mean absolute scaled errors* (MASEs) because the MASE is the recommended standard evaluation measure for time series forecasting, especially when the results of multiple forecasting problems are to be integrated (across multiple time series of different scales) [14], [15].

The MAE of a testing forecasting result is calculated as follows:

$$\boldsymbol{MAE = \frac{1}{tesl}\sum_{i=1}^{tesl}|tes\_q_i - ftes\_q_i|}.$$

The MAE is the most common and basic measure in time series forecasting. It is simple and intuitive, directly representing an average error value. However, the MAE is a scale-dependent measure; given only an MAE value, it is difficult to determine the real level of forecasting error that value represents. To overcome this shortcoming of the MAE, the MAPE was defined as shown below:

$$\boldsymbol{MAPE = \frac{1}{tesl}\sum_{i=1}^{tesl}\left|\frac{tes\_q_i - ftes\_q_i}{tes\_q_i}\right|}.$$

MAPE values are percentages; therefore, the MAPE represents the relative degree of forecasting error (compared with the actual observed value $\boldsymbol{tes\_q_i}$). The MAPE is a scale-independent measure; thus, it can be used to compare forecasting performance across disparate datasets. However, the MAPE still has some disadvantages, such as the fact that its value is infinite or undefined if $\boldsymbol{tes\_q_i} = 0$ for any $\boldsymbol{i}$ during the considered period; thus, the scale-free MASE measure was proposed by the authors of [14], [15] to avoid the disadvantages of the MAE and MAPE.

The MASE for a set of testing forecasts is calculated as follows:

$$\boldsymbol{MASE = \frac{1}{tesl}\sum_{i=1}^{tesl}\left|\frac{tes\_q_i - ftes\_q_i}{\frac{1}{tral}\sum_{j=2}^{tral}|tra\_q_j - tra\_q_{j-1}|}\right|}.$$

The MASE is similar to the MAPE; the main difference between them is that the MASE is calculated by using the training (in-sample) MAE value obtained via the naïve approach (introduced in Section 5) as the denominator in place of the actual observed value $\boldsymbol{tes\_q_i}$ that is used to calculate the MAPE.

Regarding the training MAE and MAPE measures, they can be calculated in the same way as the testing MAE and MAPE measures, respectively, with the only difference being that the inputs to these measures are $\boldsymbol{TraQoS_{1...tral}}$ and $\boldsymbol{FTraQoS_{1...tral}}$.

4. Computational Intelligence Techniques

In this paper, we regard each considered approach as consisting of two components, namely, *the predictor model/form* and *the training/fitting process*. Because of space limitations, except for the two proposed approaches (GP and SVR), we primarily discuss the former (i.e., the predictor model/form). Moreover, the GP method has no fixed, specific predictor form; thus, when introducing GP, we instead discuss its evolution (search) process.

4.1. Genetic Programming (GP)

GP is a variant of the genetic algorithm (GA) approach, and both techniques follow a similar problem-solving (evolution) process. However, they differ in the nature of the elements (solutions) that are sought and operated on during that evolution process. GAs evolve and search for a direct answer to the problem of interest (e.g., a set of values for a parameter optimization problem); by contrast, GP looks for a *solver* (in this paper, a dynamic QoS predictor, i.e., ***QoSPredictor*** as defined in Section 3.1) that can provide an answer for a given problem instance.

GP differs from the other approaches considered in this study in that it does not assume any specific form of the solver (predictor) that is sought and thus is not restricted by any default assumptions regarding the construction of the solver/predictor during its evolution process. In fact, based on the defined function set (the set of allowed expression operators) and terminal set (the set of allowed operands), GP searches for optimal solvers within the space consisting of all valid expression combinations (solvers). For example, suppose that the function and terminal sets are {+, -, *, /} and {1~10, q_{cur}, q_{cur-1}, q_{cur-2}}, respectively, where ***cur*** represents the present, the one-step-ahead forecasting target is $\boldsymbol{q_{cur+1}}$ (the prediction thereof is represented by $\boldsymbol{f_{cur+1}}$), and $\boldsymbol{q_{cur}}$, $\boldsymbol{q_{cur-1}}$, and $\boldsymbol{q_{cur-2}}$ are the lagged values (i.e., the available past observations); then, all valid expression combinations consisting of the operators and operands that appear in these sets, such as $f_{cur+1} = ((3q_{cur-1}/4.32) - (q_{cur} + q_{cur-2})*3.12)$, which is impossible to consider in other forecasting approaches, are contained in the explored search space. By contrast, as presented in the following sections, each of the other forecasting approaches employs a certain expression (predictor) skeleton; consequently, the only elements that can be modified are the parameters of that skeleton, thereby placing a considerable restriction on the expressions/predictors that can be considered. Theoretically, with GP, all possible valid predictor/expression combinations are covered (contained in the search space), and thus, this approach is more powerful and complete than any other forecasting approach because for the other approaches, the search space is limited by the default predictor/expression skeleton; the optimal solver could possess an unusual or complicated expression structure lying just outside the search space, in which case it would not be found.

Furthermore, instead of performing a complete, brute-force search in an enormous space whose size is determined by the numbers of operators and operands in the corresponding function and terminal sets, the GP algorithm executes a heuristic search driven by its fitness function and three bio-inspired GP operators, i.e., selection, crossover, and mutation (the last two operators are each associated with a probability for determining whether the corresponding operation is to be executed in each possible instance). We present a succinct, pseudocode-like description of GP below.

1. ***Initialize*** randomly *CurPopulation* **(***the first generation***)**;
2. **while** stopping condition is not satisfied
3. ***Evaluate*** each *Solver* in *CurPopulation* using ***fitness function***;
4. ***Generate*** a new empty population *NewPopulation*;
5. **while** *NewPopulation* is not full
6. ***Select*** two *Parents* from *CurPopulation* based on ***fitness value***;
7. ***Crossover*** *Parents* according to ***crossover probability***;
8. ***Mutate*** first *Parent* according to ***mutation probability***;
9. ***Mutate*** second *Parent* according to ***mutation probability***;
10. ***Push*** *Parents* into *NewPopulation;*
11. *CurPopulation* ← *NewPopulation*;
12. ***Return*** *best solver* in *CurPopulation;*

In each GP generation, all candidate solvers (expressions) constitute the population and are encoded in a tree structure, as shown in Fig. 1. The population iteratively evolves throughout subsequent generations, and

once the last generation has been produced, the algorithm returns the solver with the best survivability (represented by its fitness value) in the population as the answer (solver) for the problem of interest. The evolution of the solvers in the population is driven by the four components of the GP algorithm, namely, the fitness function and its three operators. The core of the GP algorithm is the designed fitness function, which must accurately evaluate each solver and assign it a score (fitness value) that represents its ability to address the problem of interest. In this paper, we use the MAE defined in Section 3.2 as our fitness function. Based on the fitness values, the selection operator chooses solvers from among the current population to perform two subsequent genetic operations, i.e., crossover and mutation. The former exchanges parts of two selected solvers (called parents), and the latter randomly changes one node of the selected solver. Fig. 1 presents an example illustrating these operations. For a more detailed introduction to GP, readers are referred to Chapter 9 of [22].

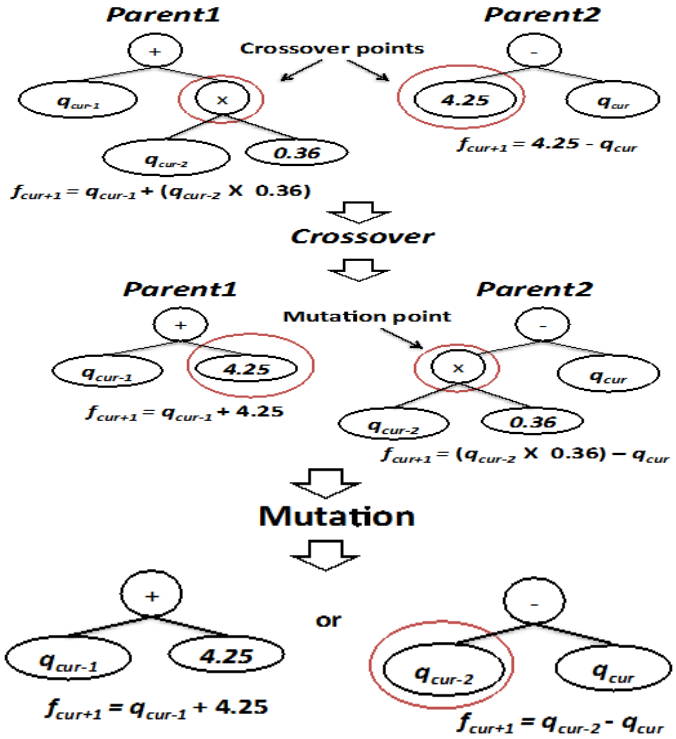

Fig. 1. A demonstration of the crossover and mutation operators.

4.2. Support Vector Regression (SVR)

Similar to the relationship between GAs and GP, SVR is a variant of the support vector machine (SVM) approach for data regression. The SVM method is a machine learning technique for data classification. An SVM algorithm classifies data into different classes through the innovative approach of mapping the data into a higher-dimensional space (feature space) based on a given function and then separating them based on a linear hyper-plane in that space [23]. The hyper-plane is linear in the higher-dimensional space but non-linear in the original feature space, thereby allowing the SVM approach to handle complex and non-linear data. In addition, instead of using all of the samples (training data) to fit the separating line or plane (the hyper-plane), which is the conventional approach in other traditional data classification techniques, SVM classification is performed based on certain selected support vectors (representative data points among the training samples).

As a variant of the SVM approach, SVR is used for data regression and thus can be used to perform time series modeling (regression) and prediction. SVR retains all of the features that characterize the SVM

technique, such as the mapping of data into a higher-dimensional feature space and the use of only selected training samples (support vectors) to perform regression [24]. We use Fig. 2 from [24] to graphically explain the SVR concept. In the figure, the blue points represent the data points and their corresponding feature vector (in our study, these data are the $\boldsymbol{q_i}$ defined in Section 3.1), the region between the two dotted lines is called the ε-insensitive tube or ε-band, and the distances from data points outside the ε-insensitive tube to the dotted lines are measured by the slack variables ξ^* and ξ.

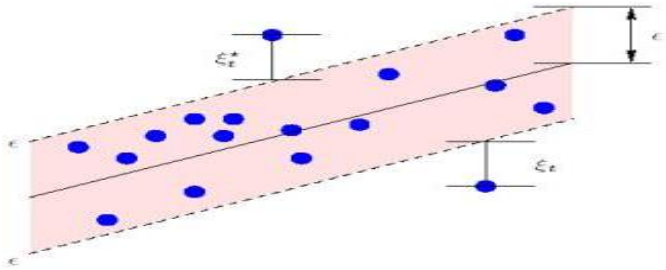

Fig. 2. Detailed illustration of the ε-band and the slack variables [24].

In SVR, the training data are first mapped into a higher-dimensional feature space using the mapping function ϕ, and then, a linear regression model (in this paper, a time series predictor) is constructed in that space; the model is defined as $\omega \cdot \phi(v) + b$, where v is the feature vector of the data points, $\phi()$ is the mapping function, and b is the bias term. Instead of using traditional loss functions, such as the least-squares function, SVR uses an innovative loss function called the ε-insensitive loss function L, which is defined as follows:

$$L = \begin{cases} 0, & if \mid q_i - (\omega \cdot \phi(v_i) + b) \mid \leq \varepsilon \\ \mid q_i - (\omega \cdot \phi(v_i) + b) \mid - \varepsilon, & otherwise \end{cases},$$

where q_i represents a real observation and $\omega \cdot \phi(v_i) + b$ represents a prediction produced by the regressed SVR model (predictor). As seen from the definition of the ε-insensitive loss function, SVR ignores any errors that are smaller than ε.

SVR can be expressed as the minimization problem defined below:

$$\min\left(\frac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{n}(\xi_i^* + \xi_i)\right)$$

$$\text{subject to} \quad \begin{cases} q_i - (\omega \cdot \phi(v_i) + b) \leq \varepsilon + \xi_i^* \\ (\omega \cdot \phi(v_i) + b) - q_i \leq \varepsilon + \xi_i, \\ \xi_i^*, \xi_i \geq 0, i = 1, \ldots, n \end{cases}$$

where n is the number of training data points and C is the cost of error. As seen from this minimization problem formulation, SVR simultaneously attempts to reduce the model complexity, represented by $\|\omega\|^2$, and to minimize the errors measured by the slack variables ξ^* and ξ. This optimization problem can be transformed into the corresponding dual problem and then solved [24].

The regression performed via SVR is different from that of traditional regression techniques (such as those introduced in Section 5.2) in that traditional regression methods do not map the data into a higher-dimensional space and lack the ability to tolerate some finite amount of error (ε is always zero) [25]. SVR is thus a more general and flexible regression approach, which can avoid over-fitting and under-fitting by means of the ε-insensitive loss function and the slack variables. More detailed introductions to SVR for time series forecasting can be found in [24]-[26].

4.3. Artificial Neural Networks (ANNs)

ANNs represent another computational intelligence (machine learning) technique that is commonly used to address problems related to diverse applications. The ANN approach has previously been applied for dynamic QoS time series forecasting by the authors of [3], [17]; thus, in this paper, we include ANNs in our comparison of forecasting approaches.

ANNs imitate the structure of the human brain and how it functions for learning and solving problems. Fig. 3 shows the internal structure of a three-layer ANN. An ANN consists of an input layer, zero or more hidden layers, and an output layer. Each layer consists of some number of neurons (e.g., *N1*, *N2*, and *N3*) connected to the neurons in the following layer (*N4*, …, and *N8*). With the exception of the neurons in the input layer (i.e., *N1*, *N2*, and *N3*), each ANN neuron has an activation function that determines whether the output connections of that neuron are activated based on weighted inputs coming from the previous layer's neurons. In an ANN, each neuron-to-neuron connection has a weight (e.g., *w14*, *w15*,…), to which a value is assigned during the training process; thus, the input value received by each neuron is a weighted combination of the values transmitted by that neuron's input connections. Finally, the last layer (the output layer) generates the final result (f_{cur+1}) of an ANN's internal calculation.

There are many different methods of training an ANN (determining its weights) for time series forecasting. Detailed introductions to ANNs as they are used in the context of time series forecasting can be found in Chapter 9.3 of [13] and in [3], [17].

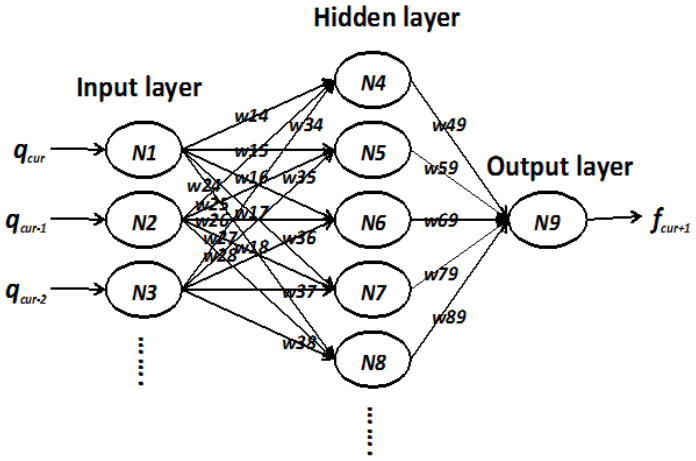

Fig. 3. The internal structure of a three-layer ANN.

5. Conventional Statistical Methods

This section introduces the conventional time series methods that are considered for comparison in this paper, with a focus on their predictor formulations (expressions). These methods can be divided into three groups, namely, *benchmark methods*, *regression methods*, and *state-of-the-art methods*. The first group consists of the baseline methods for time series forecasting. Generally, any proposed forecasting approach must at least be more accurate than the benchmark methods; otherwise, it is worthless. The second group is a set of classical regression models that are commonly used in time series forecasting. Finally, the last group consists of two sophisticated, well-developed methods that are currently used for time series forecasting. In theory, the methods in this group should be the strongest competitors for our proposed GP and SVR approaches because they are currently the de facto solutions in the field.

5.1. Benchmark Methods

Three baseline time series methods are introduced in this section.

Average method (Ave.). To generate a forecast value (f_{cur+1}), this method simply calculates the mean of all past observations ($q_{1...cur}$).

$$f_{cur+1} = \frac{1}{cur}\sum_{i=1}^{cur} q_i.$$

Naïve Method (Nai.). This method directly uses the last observed time series value (q_{cur}) as its forecast value (f_{cur+1}).

$$f_{cur+1} = q_{cur}.$$

Drift Method (Dri.). This method is a variant of the naïve method. Its forecast value is the result of the naïve method plus a drift (the second term in the expression), which is the average amount of variation observed in the historical data.

$$f_{cur+1} = q_{cur} + \frac{1}{cur-1}\sum_{i=2}^{cur} q_i - q_{i-1}.$$

A detailed introduction to these benchmark methods can be found in Chapter 2.3 of [13].

5.2. Regression Methods

Regression is a statistical technique that is used to analyze the relationships between dependent and independent variables. In this paper, we consider four different common time series regression models.

Simple linear regression (SLR). This is the simplest time series regression model:

$$f_t = \alpha + \beta \cdot (t),$$

where α is the intercept, β is the coefficient of the second term, and t is the time point to be predicted (e.g., for f_{cur+1}, $t = cur+1$).

Simple non-linear regression (SNR). This is the non-linear version of simple linear regression:

$$f_t = exp\left(\alpha + \beta \cdot \left(log(t)\right)\right),$$

where ***log()*** is the natural logarithm function and ***exp()*** is the natural exponential function. In this model, simple linear regression is used to fit a transformed linear time series that was originally non-linear (but is now linear after the natural logarithm transformation). Other transformation approaches can be applied, but the *natural logarithm approach* is common because it is simple and intuitive.

Polynomial regression (PR). The model used in polynomial regression is defined as follows:

$$f_t = \alpha + \beta_1 \cdot (t) + \beta_2 \cdot ((t)^2) + \beta_3 \cdot ((t)^3) + \ldots + \beta_i \cdot ((t)^i),$$

where α is the intercept and $\beta_1, \beta_2, \ldots$ are the coefficients of the subsequent terms.

Non-linear polynomial regression (NPR). Similar to the relationship between simple linear regression and its non-linear version, this model is the non-linear version of polynomial regression.

$$f_t = exp(\alpha + \beta_1 \cdot (log(t)) + \beta_2 \cdot (log(t)^2) + \beta_3 \cdot (log(t)^3) + \ldots + \beta_i \cdot (log(t)^i))$$

.

In regression methods, coefficients (e.g., α and β) are assigned using a selected training approach, such as the least-square-error method. Detailed descriptions of the regression models that are used in time series forecasting are presented in Chapters 4 and 5 of [13].

5.3. State-of-the-Art Methods

This section introduces the two most commonly used standard solutions in current time series research, namely, ARIMA models and ES methods.

Auto-Regressive Integrated Moving Average (ARIMA) models. These models comprise three components, namely, the *Auto-Regressive* (AR), *Integrated* (I), and *Moving Average* (MA) components. Each component is associated with an order number ($\boldsymbol{p}$ for the AR component, $\boldsymbol{d}$ for the I component, and $\boldsymbol{q}$ for the MA component) that must be determined by analyzing the past observations. ARIMA predictor models are formulated as follows:

$$f_{cur+1} = \sum_{i=0}^{p} \emptyset_i q_{cur-i} + \sum_{j=0}^{q} \theta_j \varepsilon_{cur-j} + \varepsilon_{cur+1},$$

where $\emptyset_i$ and $\boldsymbol{\theta_j}$ represent the coefficients of the predictor variables, $\boldsymbol{\varepsilon_{cur+1}}$ is the current error, and the $\boldsymbol{\varepsilon_{cur-j}}$ for $j = 0, 1, \ldots, j$ are the past errors. The first summation represents the AR component of the model, and the second summation is the MA component. The I component is a $\boldsymbol{d}$-order time differencing (pre-processing) that is applied to the predicted time series before the AR and MA components are fitted. Several well-developed methods are available for determining the values of the order numbers and the coefficients for a given forecasting problem instance, such as the Box-Jenkins algorithm and the Hyndman-Khandakar algorithm. More detailed introductions to ARIMA models can be found in [1] [2] [13] [20] [21].

Exponential Smoothing (ES) methods. ES methods constitute a group of methods, of which the most representative and complete example is the Holt-Winters method. An (additive) Holt-Winters model can be simply expressed as follows:

$$f_{cur+1} = m_{cur} + b_{cur} + c_{cur},$$

where $\boldsymbol{m_{cur}}$ describes the level of the predicted time series, $\boldsymbol{b_{cur}}$ describes its trend, and $\boldsymbol{c_{cur}}$ describes its seasonality. Because of space limitations, for the formulas for calculating the level ($\boldsymbol{m_{cur}}$), trend ($\boldsymbol{b_{cur}}$), and seasonality ($\boldsymbol{c_{cur}}$) values, readers are referred to Chapter 7.5 of [13], Section 3.1.2 of [19], and [21]. Other ES methods are variants of the Holt-Winters method that lack the trend or seasonality component (or both) or use a multiplicative model. For example, the simple exponential smoothing (SES) method considers only a level component. The ES approach implemented in this study automatically selects the most suitable ES model to use based on an analysis of the predicted QoS time series. Detailed introductions to ES can be found in Chapter 7 of [13] and [21].

6. Experiments

This section first describes our experimental environment and settings and then presents the experimental results as evaluated using the forecasting accuracy metrics defined in Section 3.2. The experimental results are discussed in detail in Section 7.

6.1. Experimental Environment

Hardware. All forecasting approaches were implemented and executed on an Apple MAC computer with a 1.3 GHz Intel Core i5 processor and 4 GB of DDR3 RAM running the OS X 10.11.6 operating system.

Programming environments. Both proposed approaches (GP and SVR) and the considered accuracy measures (the MAE, MAPE, and MASE) were implemented in Java. Our GP algorithm was implemented using a Java-based GA/GP framework called the *Java Genetic Algorithms and Genetic Programming Package* (JGAP) [27], and the SVR algorithm was implemented using a popular Java-based SVM/SVR library named the

Library for Support Vector Machines (LIBSVM) [23], [28]. The methods considered for comparison, including the ANN method, were all implemented in R using the *forecast* package [29].

QoS dataset. Our experiments were performed on the QoS dataset presented in [20], which incorporates ten distinct dynamic QoS time series collected from ten real-world WSs (e.g., Google and Amazon WSs) and has also been employed in other QoS forecasting studies, such as [1] [20]. More detailed descriptions of the dataset can be found in [1] [20]. To generate more problem instances on which to run our experiments, we split each time series into several *time series segments* (TSs). These TSs were labeled as TS1_1 (segment 1 of time series 1), TS1_2 (segment 2 of time series 1), TS2_1 (segment 1 of time series 2), etc.

Problem configuration. In this paper, the lengths (***tral*** and ***tesl***) of the training dataset (***TraQoS***) and testing dataset (***TesQoS***) were set to 500 and 100, respectively (thus, the length of each TS was 600), which means that each trained/fitted QoS predictor was used to predict the QoS values at 100 future time points. Additionally, in this study, each QoS time series in the dataset was split into 10 TSs to run our experiments.

6.2. Experimental Results

Table 2. The Training (in-sample) MAE Results

	TS1_1	TS1_2	TS1_3	TS1_4	TS1_5	TS1_6	TS1_7	TS1_8	TS1_9	TS1_10	TS2_1	TS2_2	TS2_3	TS2_4	TS2_5	TS2_6	TS2_7	TS2_8	TS2_9	TS2_10	TS3_1	TS3_2	TS3_3	TS3_4	TS3_5
GP	1880	1836	1599	1426	1170	1695	1601	1613	1708	1542	1095	1304	1205	1146	1334	1093	1112	1243	1288	1329	8871	10046	10241	10302	10932
SVR	5235	5115	4810	4619	2027	3712	3641	3676	3773	2840	2565	4565	4402	4436	4568	2903	1256	1379	1423	1465	9911	14350	14179	14210	14749
ANNs	5459	5195	5047	4578	1597	1994	2288	2429	2127	2026	3328	4658	4629	4359	4479	2793	1833	1910	2209	1759	17245	17780	17313	17614	18012
Ave.	7124	7121	7182	7293	4836	5209	5011	5000	5022	4445	2814	4963	5291	5688	6088	4769	2850	2653	2525	2370	17825	18398	18378	18979	20011
Nai.	2489	2482	2068	1820	1797	2425	2231	2250	2452	2372	1408	1638	1376	1455	1697	1611	1935	2127	2228	2233	17082	18922	18548	18649	19496
Dri.	5920	5900	5345	4962	10152	3740	3709	7660	5820	2985	37099	4604	4405	4470	4630	3828	2672	2340	1462	1650	14624	14358	14184	14362	14752
SLR	11040	7479	5448	11996	5946	11564	7329	4049	15095	14268	8010	12349	9160	5090	10466	10864	3300	3152	2025	1611	17847	37308	30550	21648	16712
SNR	6658	6045	5160	4743	2874	5197	4870	4343	3757	3974	2984	5211	5056	4854	4675	3104	1510	1673	1590	1591	11123	16245	15776	15490	15129
PR	14228	12991	20941	126758	197218	692429	259548	2376059	3451513	1240634	29564	24793	16131	153473	418242	527539	140959	234374	290459	136401	35458	116243	54347	228146	143982
NPR	6385	5480	6218	8148	14332	30923	6202	8138	4038	65142	4223	6611	4519	5103	4588	4071	7227	2260	2377	1886	14145	24125	15047	15867	15791
ARIMA	2862	2825	2347	2125	1523	1969	1785	1817	1973	1960	1497	1626	1574	1440	1699	1568	1811	1642	1759	1728	16228	16842	18609	18635	19171
ES	3291	3479	3335	2901	1906	2001	1849	1866	2054	2080	1991	2830	2706	2846	3232	1555	1517	1655	1742	1733	18212	19705	19641	19956	20380

	TS3_6	TS3_7	TS3_8	TS3_9	TS3_10	TS4_1	TS4_2	TS4_3	TS4_4	TS4_5	TS4_6	TS4_7	TS4_8	TS4_9	TS4_10	TS5_1	TS5_2	TS5_3	TS5_4	TS5_5	TS5_6	TS5_7	TS5_8	TS5_9	TS5_10
GP	8710	6185	5886	6487	5824	1564	1607	1661	1378	1536	2835	2603	2459	2440	2346	610	672	675	675	750	807	789	793	890	810
SVR	12007	6429	5965	6567	5864	3078	3870	3950	2270	2417	7220	6249	6073	6067	5936	663	767	784	789	870	887	817	842	953	884
ANNs	25320	22196	10275	12063	10403	4247	4583	4234	2043	2254	11620	11074	10925	11010	6642	713	812	839	849	1000	1089	988	1030	1173	1085
Ave.	19306	16387	15234	15142	13991	4215	4827	5198	3709	3602	8564	8389	8474	8639	8713	727	808	852	886	967	1011	1027	1080	1189	1173
Nai.	16368	12137	11156	12166	10946	2032	1989	2070	1962	2222	4829	4685	4489	4468	4257	923	1004	1001	1004	1096	1191	1150	1196	1323	1245
Dri.	12019	6638	5979	6625	5899	11441	4717	4251	7471	2450	8114	6255	6090	6227	6039	1213	778	893	817	986	1041	4202	845	956	910
SLR	26340	8896	6616	12423	9977	3979	4893	4232	3000	2538	22559	16213	8662	27019	50356	759	1009	890	931	899	911	1219	924	1360	1027
SNR	12297	6965	6364	7109	6349	3249	4197	4069	2504	2620	7223	6448	6203	6128	5951	694	829	814	848	896	885	921	917	1016	951
PR	1149915	46593	101737	154828	404383	8106	6923	28743	64377	86850	1691478	482581	3735974	7543606	23661414	1177	1099	2093	13980	29877	5142	1509	14068	6817	18480
NPR	16181	7194	6114	8890	6010	3219	4269	4630	2516	2645	8837	6278	6485	6177	6944	705	1027	803	845	875	1606	951	974	1010	911
ARIMA	15580	11065	10100	11135	9728	2203	2226	2348	1786	2002	8830	9029	8326	7949	6805	723	800	827	836	933	1002	977	1030	1182	1106
ES	17116	16387	15233	12242	10562	2449	2429	2830	1778	2005	9508	9262	8080	9195	8355	727	877	834	846	1012	1074	1018	1050	1266	1096

Table 3. The Testing (out-of-sample) MAE Results

	TS1_1	TS1_2	TS1_3	TS1_4	TS1_5	TS1_6	TS1_7	TS1_8	TS1_9	TS1_10	TS2_1	TS2_2	TS2_3	TS2_4	TS2_5	TS2_6	TS2_7	TS2_8	TS2_9	TS2_10	TS3_1	TS3_2	TS3_3	TS3_4	TS3_5
GP	1389	893	762	2698	4503	780	983	1381	1964	961	2307	854	1170	1454	1060	2314	1580	1123	1473	1460	29453	14525	6947	7721	4227
SVR	1141	828	645	6409	9596	808	1023	1132	1739	906	10795	788	863	1135	975	2631	1404	1104	1366	1332	34986	6710	6780	7543	4066
ANNs	3237	1377	1602	5352	5090	867	1305	2246	2099	1223	10493	1454	2417	2543	1669	2490	2242	1699	1681	1592	29852	15639	14771	14419	11858
Ave.	4856	4114	4002	6214	8319	3197	3161	3263	3248	2228	11961	3795	3843	3837	3828	2406	1597	1661	1632	1607	30067	18303	17750	19136	15618
Nai.	2041	1261	954	2876	4991	1074	1355	1964	2475	1110	1983	1091	1397	1941	1642	3603	2054	1898	1968	1599	34084	13084	13127	13798	7749
Dri.	2219	1913	1521	6803	6364	743	921	8474	4853	1051	28847	763	846	1160	986	2078	2484	2455	1403	1500	30965	6738	6765	7631	4063
SLR	10191	3944	1251	12473	7178	10084	5748	2165	3737	2806	14175	11687	6906	2169	4016	6369	2218	2352	1707	1558	29986	39405	29187	20293	6083
SNR	3445	2109	1241	6533	8632	3884	3242	2456	1883	1149	10889	1811	1819	1577	1177	2790	1473	1502	1479	1416	33725	11244	10033	10448	5097
PR	43664	31198	16670	36562	28400	28213	7023	34559	20463	6401	72854	67628	4136	40296	34527	18311	7201	4013	1944	2968	53262	228406	45279	55964	28347
NPR	2668	1814	2134	5665	7261	12318	1262	2987	2217	2990	12347	4394	903	1696	975	2108	2190	1747	1388	1336	30719	28638	8653	8869	4680
ARIMA	1843	1027	826	2599	4231	908	1118	1603	2120	1087	3157	1070	1523	1869	1514	3153	2135	1580	1636	1553	30262	21502	14914	15130	11237
ES	1767	905	823	2380	4342	899	1173	1580	2111	1237	7213	1058	1266	1541	1327	3392	2016	1496	1571	1573	33913	15596	14028	14574	8031

	TS3_6	TS3_7	TS3_8	TS3_9	TS3_10	TS4_1	TS4_2	TS4_3	TS4_4	TS4_5	TS4_6	TS4_7	TS4_8	TS4_9	TS4_10	TS5_1	TS5_2	TS5_3	TS5_4	TS5_5	TS5_6	TS5_7	TS5_8	TS5_9	TS5_10
GP	7812	4377	9947	4157	5411	1933	2285	2106	1683	27816	942	966	1975	1102	1450	1131	671	828	1054	909	888	679	1360	603	2479
SVR	7128	4356	9787	4039	5253	5582	1886	1642	1615	25406	853	901	1586	960	1198	1150	702	821	974	849	912	758	1342	628	2447
ANNs	13615	7482	19722	7760	9548	4220	3656	2640	2154	27330	2431	2400	2864	2018	1465	1025	835	924	1250	1024	1172	995	1525	932	2821
Ave.	16047	8727	13420	8721	8984	5174	3820	3429	2381	29137	5924	4946	4986	4559	4503	1048	914	995	1187	977	1160	1120	1526	1107	2957
Nai.	12928	8176	18180	7697	9985	2176	2490	2726	2515	14237	1459	1507	2623	1459	2214	1315	897	1125	1391	1226	1112	1129	1759	998	4636
Dri.	7156	4478	9827	4119	5276	6771	3230	1777	7570	25443	2532	891	1639	1204	1322	1188	663	814	948	1011	918	4523	1339	607	2445
SLR	10853	7667	11368	9326	9016	5126	3430	1763	2489	25600	14301	8615	2310	5127	12112	1036	1206	1030	1220	916	1071	1274	1472	1219	2789
SNR	7160	4736	9981	4642	5687	5434	2430	1871	1936	25634	897	1200	1626	1006	1193	1059	887	905	1101	883	957	1008	1420	853	2594
PR	53062	10666	11069	9208	11506	31601	16355	22581	16474	30464	85486	37311	54990	46165	99594	2175	1381	1930	4269	2859	1012	1182	1660	1737	2479
NPR	8128	4826	9852	5079	5559	6017	2639	2508	1653	25423	1736	1071	1606	1007	1220	1052	1321	887	941	851	1200	1031	1351	890	2561
ARIMA	12762	8813	13167	8819	8935	2016	2779	2712	2044	33149	3775	4098	8273	3639	4421	1048	840	974	1202	967	1105	1081	1553	1034	3334
ES	13693	8727	13419	8972	8756	1981	2575	2093	2094	36380	1294	1427	2694	1438	1623	1048	782	902	1250	1049	1092	945	1731	935	4436

In this section, we present the training (in-sample) and testing (out-of-sample) results obtained in the experiments. For both types of results, the MAE and MAPE measures were calculated, and for the testing accuracy, the MASE was also considered; all measures are defined in Section 3.2. However, because of space limitations, we show only the MAE values for each individual TS experiment. Regarding the MAPE and MASE results, because they are both scale-independent and can be compared across different TSs [14], [15], we present only the average (mean) results for all experiments.

Table 4 lists the training MAE results for the experiments, and the testing MAE results are shown in Table 3. The average results for all experiments are presented in Table 4. For each individual TS experiment in Table 2 and Table 3 and each average result in Table 4, the value corresponding to the best approach (the lowest value) is shown in red, the second-best approach is indicated in blue, and the third-best approach is indicated in green.

Table 4. Average Training and Testing Results

	Training (in-sample)		*Testing (Out-of-sample)*		
	MAE	*MAPE*	*MAE*	*MAPE*	*MASE*
GP	2792	0.23	3571	0.374	1.001
SVR	4561	0.248	3790	0.247	1.197
ANNs	6223	1.03	5221	0.885	1.423
Ave.	6919	1.135	6348	1.23	1.809
Nai.	4794	0.698	4542	0.732	1.141
Dri.	6170	1.024	4625	0.841	1.718
SLR	10039	2.19	7601	1.603	2.219
SNR	5002	0.452	4363	0.473	1.373
PR	1004084	357.59	29510	8.898	9.098
NPR	7779	1.036	4847	0.639	1.494
ARIMA	4911	0.866	5243	0.998	1.312
ES	5634	0.936	4943	0.855	1.321

7. Discussion

This section discusses the experimental results (i.e., Table 2, Table 3, and Table 4) reported in Section 6.2. We analyze first the training performance and then the testing performance achieved in the experiments. From Table 2, it is obvious that GP is superior for all TSs in terms of the training MAE results (i.e., it yields the lowest MAE value in each TS experiment). In addition, Table 4 shows that GP achieves the lowest average values of both the training MAE and the training MAPE. In our experiments, GP is clearly the best-performing approach during the training stage for dynamic QoS time series forecasting. Thus, if the purpose is to accurately model and fit a known QoS time series, GP appears to be the best choice. Regarding the second-best-performing approach during the training stage, Table 2 indicates several competitors, including SVR (which yields the second-lowest MAE value on 23 TSs), Nai. (the second-lowest value on 15 TSs), and ARIMA (the second-lowest value on 9 TSs). Thus, SVR may be considered the winner here because it achieves the second-lowest MAE value on more TSs than either of its two competitors. Moreover, as seen from the average training results reported in Table 4, SVR yields the second-lowest value for both considered measures. These observations suggest that during the training stage, on average, SVR is the second-best-performing approach in our experiments.

Regarding training performance, the results reported in Table 2 and Table 4 show that in a comparison between the two proposed computational intelligence techniques (GP and SVR) and all time series approaches considered for comparison, including the two current de facto approaches (ARIMA and ES), the former methods clearly outperform the latter. The experimental results demonstrate the potential of GP and SVR for fitting, modeling, and describing dynamic QoS time series, which can be used to predict the QoS

performance of a WS.

Regarding the testing accuracy, the results are more complicated because no single approach clearly dominates (achieves the lowest value on each TS and in each measure) according to Table 3 and Table 4. Below, we first present a comparison between the two proposed computational intelligence techniques and then compare these techniques with the conventional time series approaches. In Table 3, SVR achieves the lowest MAE value on 26 TSs; Dri., on 12 TSs; and GP, on 5 TSs. It seems that in contrast to the training results presented in Table 3, GP is inferior to SVR in terms of the testing results shown in Table 3. However, although GP shows the best testing performance on fewer TSs than does SVR, it still yields the best average testing accuracy according to two of the three considered measures (i.e., the MAE and MASE), as seen in Table 4. Among the testing MAE results presented in Table 3, in many of the TS experiments in which SVR outperforms GP, such as TS1_2, TS1_10, TS2_2, TS2_5, TS2_8, TS3_7, TS4_4, TS4_6, TS4_7, TS5_3, TS5_4, TS5_5, TS5_8, and TS5_10, the difference is not significant. However, in experiments TS1_4, TS1_5, TS2_1, and TS4_1, in which GP outperforms SVR in terms of testing performance, SVR yields an MAE value that is two or nearly three times worse than that achieved by GP. This may be the reason why GP outperforms SVR in two of the three average testing measures in Table 4, despite the fact that GP yields the best MAE in significantly fewer experiments. Overall, the superiority of SVR over GP is minor (insignificant) in many cases, whereas the inferiority of SVR compared with GP in some cases is major (significant). Based on the above observations, between GP and SVR, we consider that GP demonstrates a more stable and reliable forecasting capability. An easy way to compare the two proposed computational intelligence techniques with the conventional time series approaches is to consider the average testing accuracy information presented in Table 4. According to this table, both GP and SVR are among the top three best-performing forecasting approaches in terms of all three accuracy measures. In fact, GP and SVR are the top two approaches in all cases except for the average testing MASE, for which SVR is only the third-best-performing approach. Thus, on average, the proposed computational intelligence techniques clearly outperform the other approaches considered for comparison.

Although GP may be the most reasonable choice overall for dynamic QoS time series forecasting, a major cost of using this approach is that GP requires more time to generate a predictor for a given forecasting problem instance because of the time-consuming nature of the evolution process. However, if the forecasting quality is the only concern, GP is still the recommended approach. Otherwise, enhancing the available computation resources and hardware specifications can reduce the time consumed by GP. This problem can also be relieved by using parallel processing platforms provided through the cloud.

8. Conclusion

Recently, the prediction of future values of dynamic QoS attributes for WSs has become an important emerging topic in services computing. In most previous studies, traditional time series approaches have been used to address this problem. However, this paper demonstrates that in dynamic QoS time series forecasting, computational intelligence techniques (i.e., GP and SVR) can outperform traditional time series methods in terms of both training (in-sample) and testing (out-of-sample) forecasting accuracy. According to our experimental results, when the purpose is to model and fit a known QoS time series, GP is the superior choice because it offers the best training forecasting accuracy, both on average and for individual problem instances. For the prediction of unknown future QoS values, overall, both of the proposed computational intelligence techniques outperform the other approaches considered for comparison. Based on a comparison of the testing performances of GP and SVM, we consider that GP may be the better choice because it demonstrates better average forecasting accuracy and performs only slightly worse than SVM in many cases, whereas SVM, although it yields the lowest forecasting error in many cases, also yields

significantly higher forecasting errors in several cases, indicating that its performance is unstable and unreliable.

In this paper, we have demonstrated that techniques developed in the field of computational intelligence outperform even the most sophisticated traditional methods of dynamic QoS time series forecasting in services computing. To some extent, this paper can be viewed as a case study of the competition between computational intelligence and traditional disciplines (in this case, statistics and time series forecasting). According to the experimental results presented in this paper, computational intelligence clearly wins the competition in this case.

In the future, we plan to apply computational intelligence techniques to other time series application domains, such as the forecasting of cloud workloads and mobile traffic. Another goal is to enhance the forecasting accuracy of such techniques while simultaneously reducing their cost.

Acknowledgment

This research is partially sponsored by the Ministry of Science and Technology (Taiwan) under the Grant MOST106-2221-E-030-002, MOST106-2811-E-001-003, and MOST106-2221-E-001-007-MY2.

References

[1] Amin, A., Colman, A., & Grunske, L. (2012). An approach to forecasting QoS attributes of web services based on ARIMA and GARCH models. *Proceedings of 2012 IEEE 19th International Conference on Web Services (ICWS)* (pp. 74-81).

[2] Amin, A., Grunske, L., & Colman, A. (2012). An automated approach to forecasting QoS attributes based on linear and non-linear time series modeling. *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering, Essen, Germany.*

[3] Zadeh, M. H., & seyyedi, M. (2010). Qos monitoring for web services by time series forecasting. *Proceedings of the 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT)* (pp. 659-663).

[4] Huiyuan, Z., Weiliang, Z., Jian, Y., & Bouguettaya, A. (2013). QoS analysis for web service compositions with complex structures. *Services Computing, IEEE Transactions on, 6(3),* 373-386.

[5] YunNi, X., Jian, D., Xin, L., & QingSheng, Z. (2013). Dependability prediction of WS-BPEL service compositions using petri net and time series models. *Proceedings of IEEE 7th International Symposium on Service Oriented System Engineering (SOSE)* (pp. 192-202).

[6] Yilei, Z., Zibin, Z., & Lyu, M. R. (2011). WSPred: A time-aware personalized QoS prediction framework for web services. *Proceedings of IEEE 22nd International Symposium on Software Reliability Engineering (ISSRE)* (pp. 210-219).

[7] Godse, M., Bellur, U., & Sonar, R. (2010). Automating QoS based service selection. *Proceedings of IEEE International Conference on Web Services (ICWS)* (pp. 534-541).

[8] Iba, H., & Nikolaev, N. (2000). Genetic programming polynomial models of financial data series. *Proceedings of the 2000 Congress on Evolutionary Computation* (pp. 1459-1466).

[9] Iba, H., & T. Sasaki. (1999). Using genetic programming to predict financial data. *Proceedings of the 1999 Congress on Evolutionary Computation* (p. 251).

[10] Santini, M., & Tettamanzi, A. (2001). Genetic programming for financial time series prediction. *Genetic Programming*, 361-370.

[11] Wagner, N., Michalewicz, Z., Khouja, M., & McGregor, R. R. (2007). Time series forecasting for dynamic environments: The DyFor genetic program model. *IEEE Transactions on Evolutionary Computation, 11(4),* 433-452.

[12] Agapitos, A., Dyson, M., Kovalchuk, J., & Lucas, S. M. (2008). On the genetic programming of time-series predictors for supply chain management. *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation* (pp. 1163-1170).

[13] Hyndman, R. J., & Athanasopoulos, G. (2014). Forecasting: Principles and practice. *OTexts.*

[14] Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting, 22(4),* 679-688.

[15] Hyndman, R. J. *Another Look at Forecast-Accuracy Metrics for Intermittent Demand.*

[16] Tashman, L. J. (2000). Out-of-sample tests of forecasting accuracy: An analysis and review. *International Journal of Forecasting, 16(4),* 437-450.

[17] Senivongse, T., & Wongsawangpanich, N. (2011). Composing Services of different granularity and varying QoS using genetic algorithm. *Proceedings of the World Congress on Engineering and Computer Science Lecture Notes in Engineering and Computer Science,* San Francisco (pp. 388-393).

[18] Mu, L., Jinpeng, H., & Huipeng, G. (2009). An adaptive web services selection method based on the QoS prediction mechanism. *Proceedings of IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technologie* (pp. 395-402).

[19] Ye, Z., Mistry, S. K., Bouguettaya, A., & Dong, H. (2014). Long-term QoS-aware cloud service composition using multivariate time series analysis. *IEEE Transactions on Services Computing, 99.*

[20] Cavallo, B. Penta, M. D., & Canfora, G. (2010). An empirical comparison of methods to support QoS-aware service selection. *Proceedings of the 2nd International Workshop on Principles of Engineering Service-Oriented Systems.*

[21] Rahman, Z. U., Hussain, O. K., & Hussain, F. K. (2014). Time series QoS forecasting for management of cloud services. *Proceedings of the 2014 Ninth International Conference on Broadband and Wireless Computing, Communication and Applications.*

[22] Mitchell, T. (1997). *Machine Learning*. McGraw Hill.

[23] Hsu, C. W., Chang, C. C., & Lin, C. J. (2003). *A Practical Guide to Support Vector Classification.*

[24] *Support Vector Machine Regression*. Retrieved from http://kernelsvm.tripod.com/

[25] Chen, B. J., Chang, M. W., & Chih-Jen, L. (2004). Load forecasting using support vector machines: A study on EUNITE competition 2001. *IEEE Transactions on Power Systems, 19(4),* 1821-1830.

[26] Müller, K. R., Smola, A. J., Rätsch, G., Schölkopf, B., Kohlmorgen, J., & Vapnik, V. (1997). Predicting time series with support vector machines. *Proceedings of International Conference on Artificial Neural Networks* (pp. 999-1004).

[27] Meffert. K. *JGAP - Java Genetic Algorithms and Genetic Programming Package*. Retrieved from http://jgap.sf.net/

[28] Chang, C. C., & Lin, C. J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST), 2(3),* 27.

[29] Hyndman, R. J. (2015). *Forecasting Functions for Time Series and Linear Models*. Retrieved from http://github.com/robjhyndman/forecast

Yang Syu received his B.S degree in computer science and information engineering from Fu-Jen Catholic University, Taiwan, in 2010, and his Ph.D degree from the National Taipei University of Technology, Taiwan, in 2017. He is now a postdoctoral fellow at Institute of Information Science, Academia Sinica in Taiwan. His research interests include service-oriented computing, semantic web, and mobile computing.

Yong-Yi Fanjiang received his MS and Ph.D degree in computer science and information engineering from National Chiao Tung University, Taiwan, in 1998 and from National Central University, Taiwan, in 2004, respectively. Currently, he is an associate professor in the Department of Computer Science and Information Engineering and the chair in the bachelor's program in software engineering and digital innovation applications, Fu Jen Catholic University, Taiwan. His research interests include mobile and pervasive computing, software engineering, semantic web, and human computer interaction.